Scientific Computing with (Short version)

Rafael Palacios (Feb/2014)

Contents

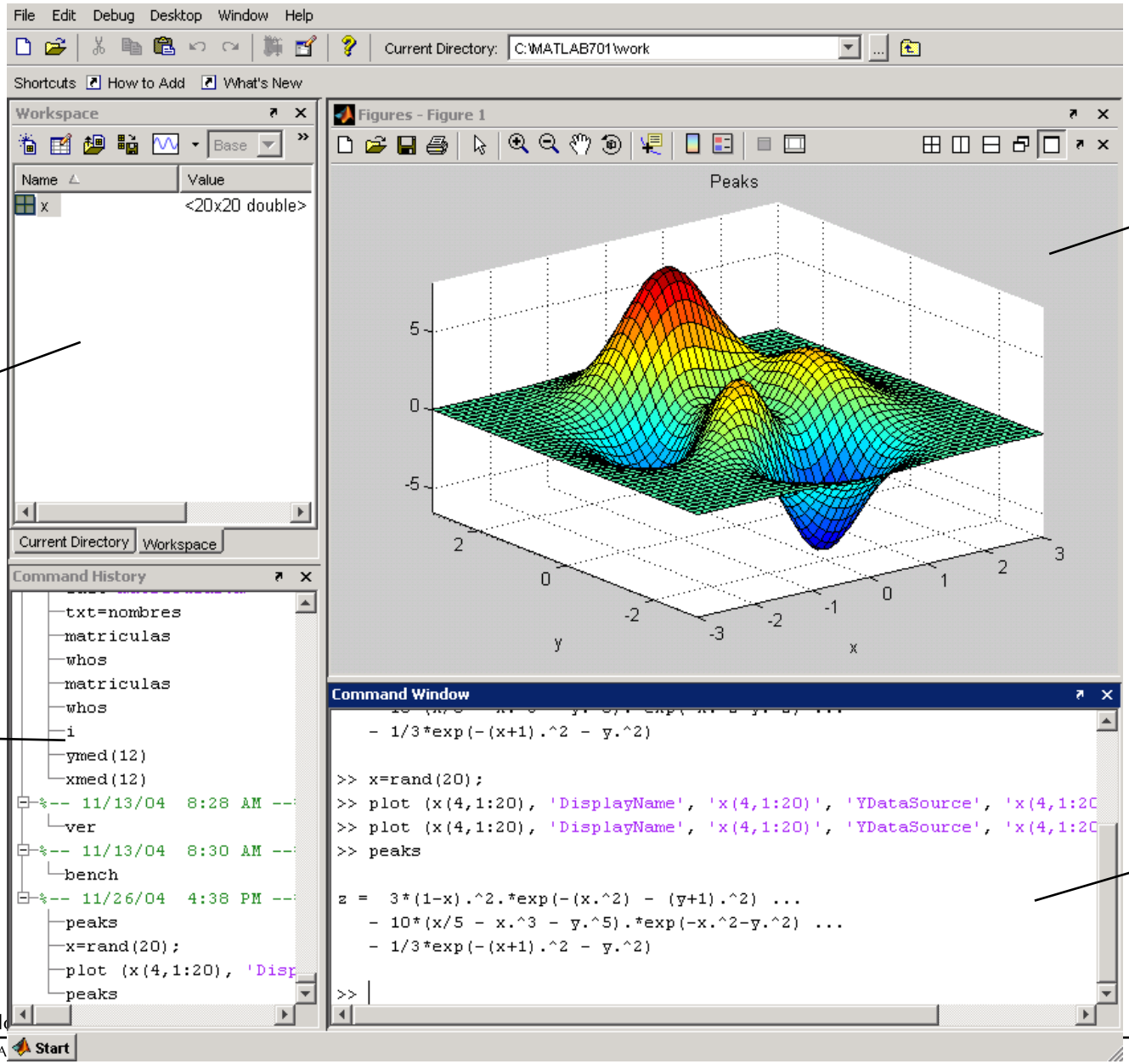
1. Introduction to Matlab.
2. Basic data types.
3. Matlab programming.
4. Advanced data types.
5. Code Optimization.
6. Graphical representation.
7. Developing standalone applications.

Contents

Introduction to Matlab

- ¿What is Matlab?
  - Matlab = **Mat**rix **Lab**oratory.
  - Interactive program for mathematical operations and graphical representation
  - Company: The Mathworks Inc (Natick, MA). `http://www.mathworks.com`
  - Created in California by Jack Little and Cleve Moler in 1984, for matrix computation without the need of background in programming (no loops).

Matlab 7 development environment



Workspace

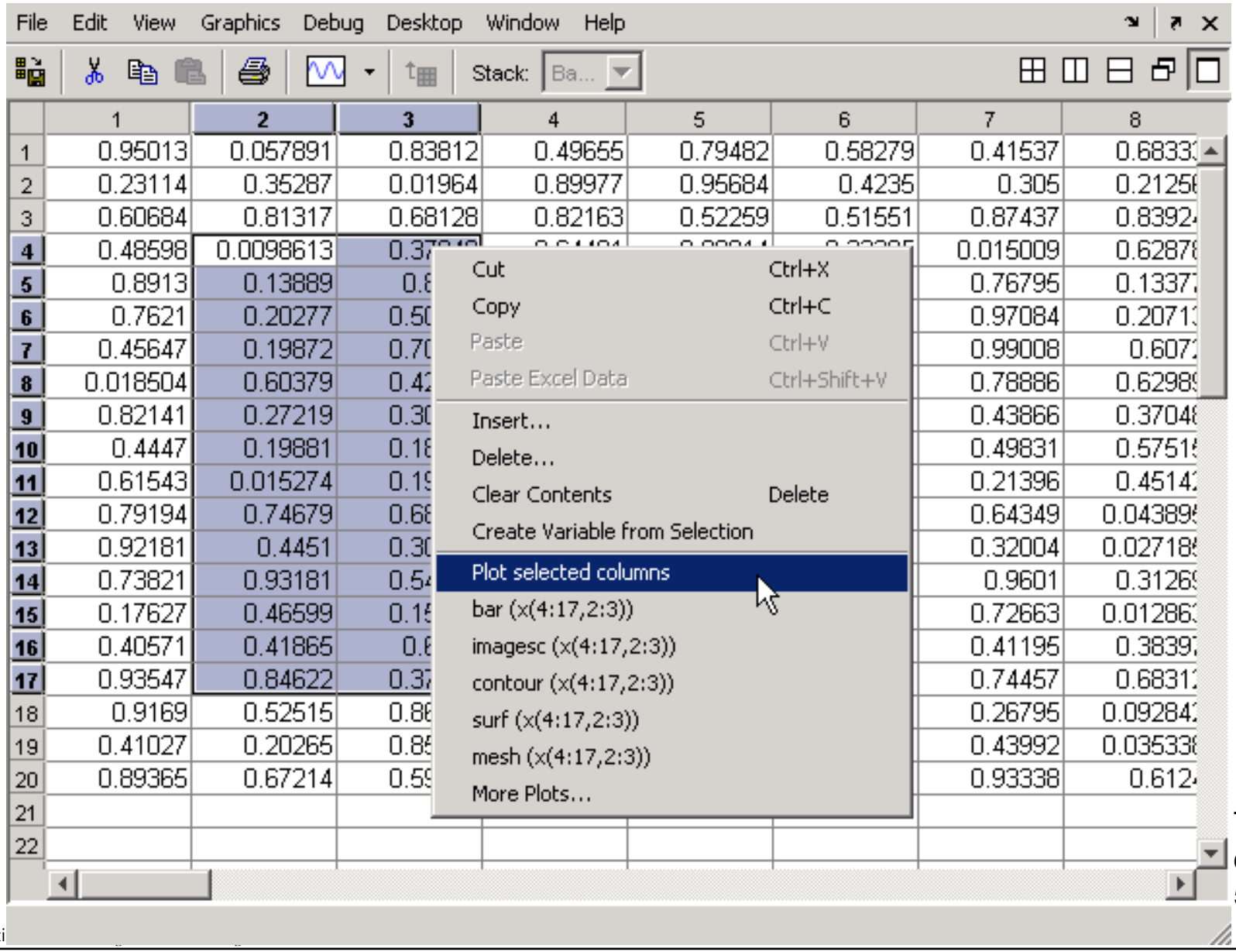
command history

Figuras

command window

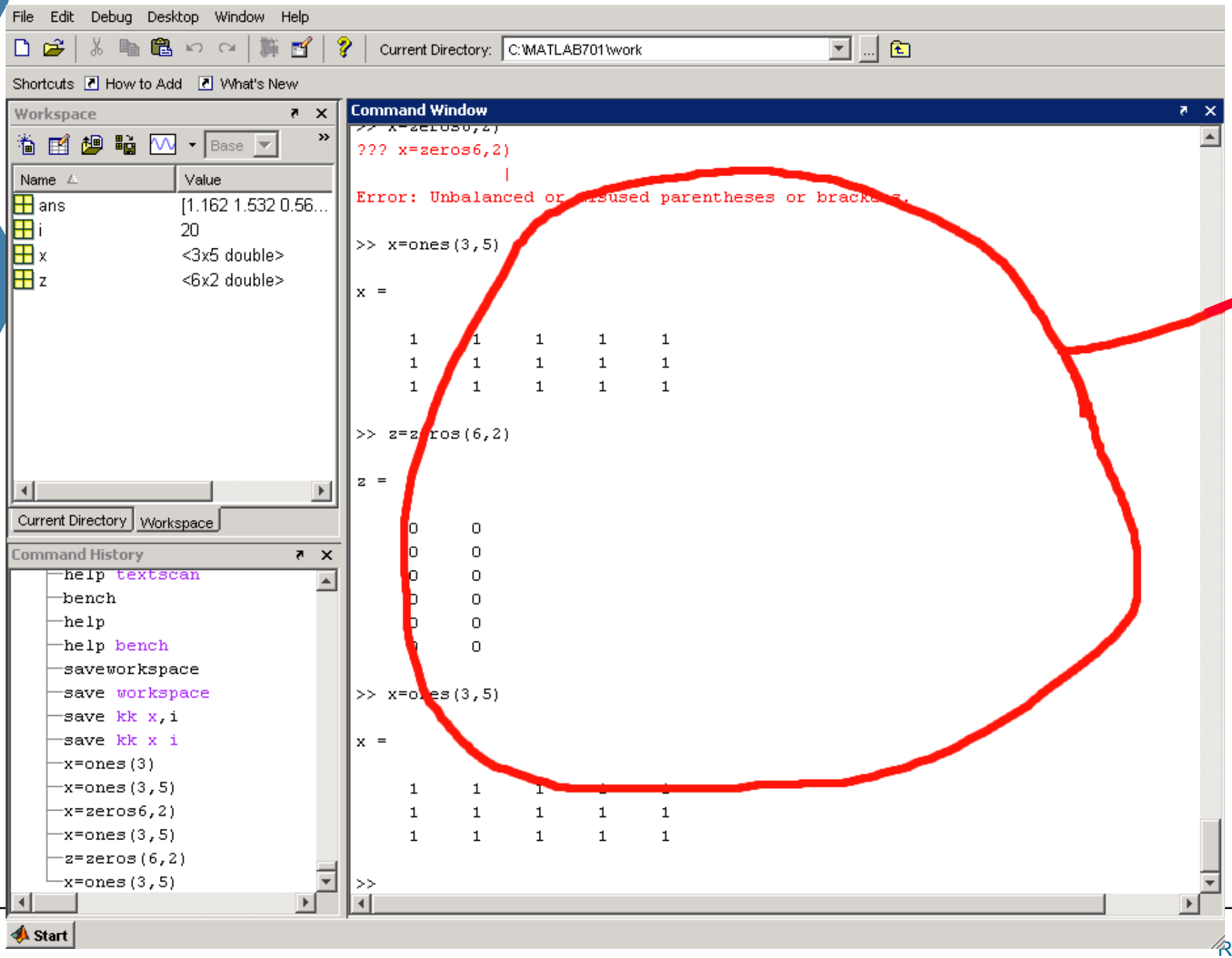
Algunas mejoras de Matlab 7

- Immediate data plot



Tamaño máximo de la matriz: 524288 elementos

Development environment



Command window

Basic commands

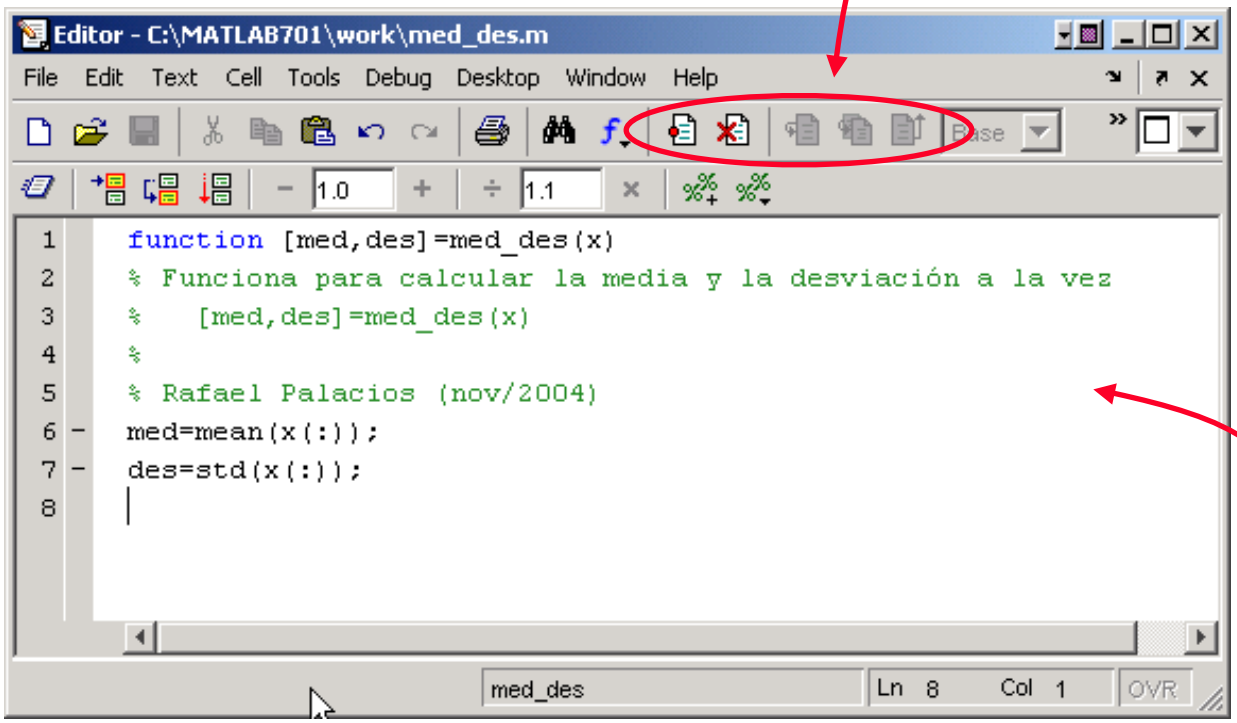
- `ver` →version number, license code and toolbox versions
  - License 46431: Research
  - License 205966: Only for teaching
- `whos` →list of variables
- `save archivo` →save all variables
- `save archivo a b` →save variables `a` & `b`
- `load archivo` → load file and create variables
- `quit` →exit

Editor

- Matlab includes and editor for writing programs and functions

Controles del debugger

Ejecución por secciones en cell mode

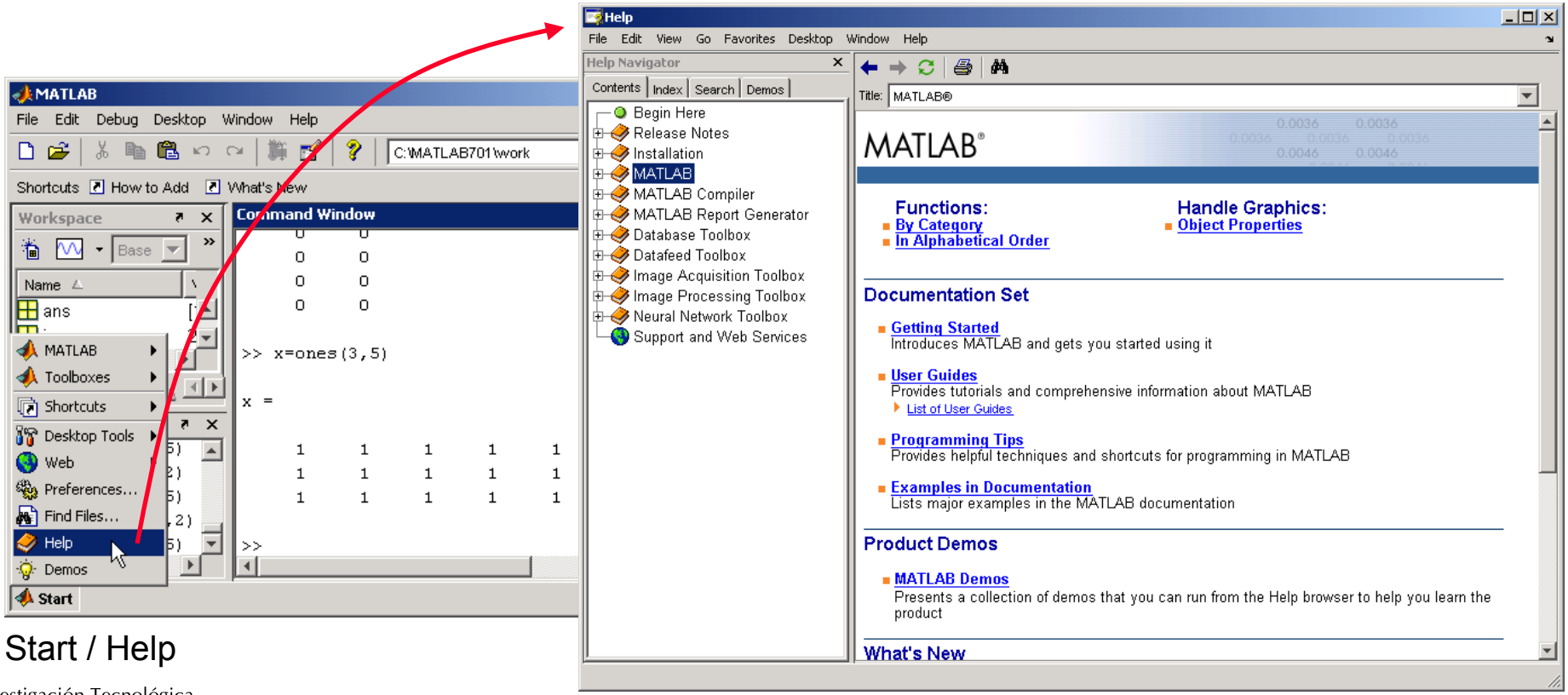


sintaxis

Documentation

- Ayuda en modo texto mediante comandos
  - `help función` →muestra la ayuda de una función
  - `help` vale tanto para las funciones del sistema como para desarrollos propios
- Ayuda de tipo gráfico

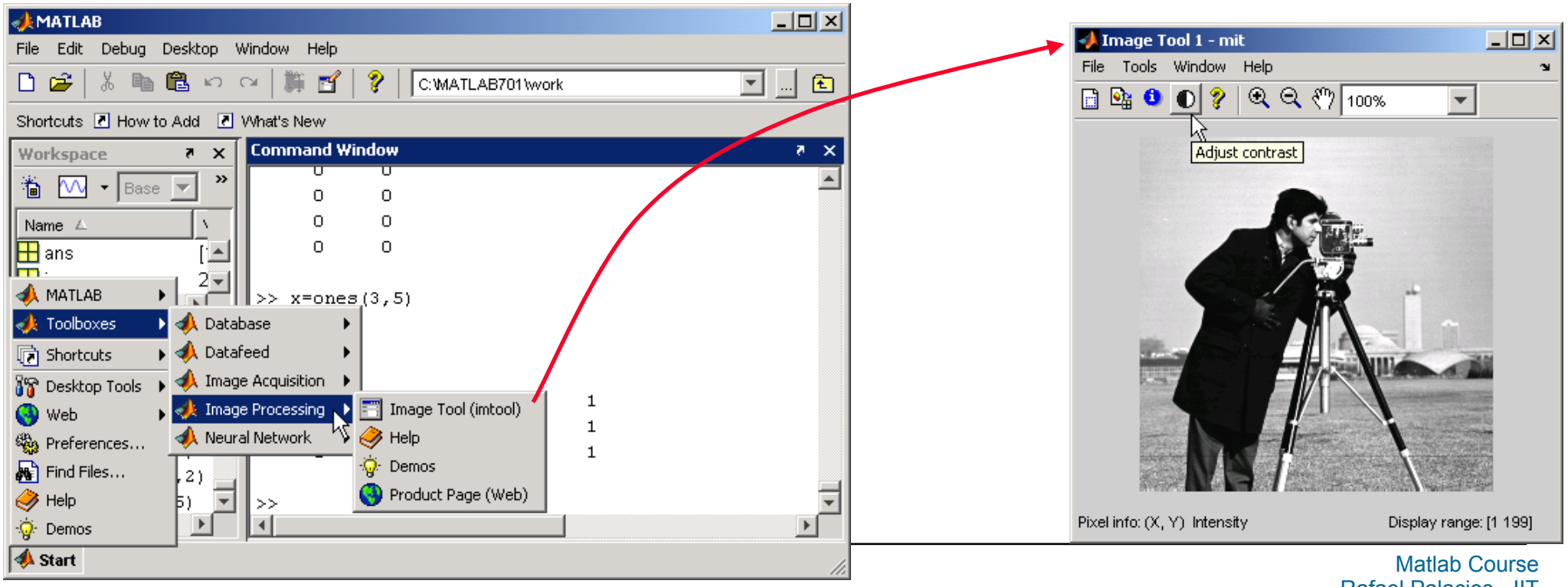
Hypertext Help Window



Start / Help

Toolboxes

- Specific libraries for different scientific topics. Include:
  - User's Guide [HTML, PDF]
  - Reference Guide [HTML, PDF]
  - Demo Programs
  - Demo applictions (called tool) ready to use



Try it yourself

- Everybody
  - Open Matlab
  - Identify all sections of the environment
  - Get the list of toolboxes installed
  - Declare a matrix, ex: a(3,4)=25;
  - Edit values of the matrix
  - Plot columns/rows
- Advanced
  - Plot surface. Edit surface.
  - Use a demo program of one toolbox

Contents

1. Introduction to Matlab.
2. Basic data types.
3. Matlab programming.
4. Advanced data types.
5. Code Optimization.
6. Graphical representation.
7. Developing standalone applications.

Variables

- Matlab doesn't require to declare variable or to specify the size of arrays
  - Variables are auto-declared when assigned
  - Memory is reallocated automatically

```
>> x=5;
>> y=20;
>> z=x*y

z =

   100

>> a=load('data.txt');
>> st='hola';
```

using ';' the value is assigned but the result is not diplayed

without ';' the final result is shown

Vectors and Matrices

- Matlab considers that all variables are matrices
- Vectores and scalars are special cases (size 1).

Some examples on how to initialize **row vectors**

```
>> x=[1,2,3,5,7,11,13];  → [ 1  2  3  5  7  11  13 ]
>> x=[1 2 3 5 7 11 13];  → [ 1  2  3  5  7  11  13 ]

>> y=1:5;                → [ 1  2  3  4  5 ]
>> even=2:2:10;          → [ 2  4  6  8  10 ]
>> odd_down=9:-2:1;      → [ 9  7  5  3  1 ]

>>a(5)=7;                 → [ 0  0  0  0  7 ]
```

Vectors and Matrices

Examples on how to initialize matrices

```
>> M = [1 2 3; 4 5 6; 7 8 9];

>> ceros=zeros(2,5);

>> unos=ones(3,4);

>> M2=[ 20, 21, 22;  M];
>> M2=[[20, 21, 22]; M];

>> M3=[ [15;16;17], M];

>> aleatorio=rand(20,30);
>> normal=randn(20,30);
```

```
1     2     3
4     5     6
7     8     9
```

```
0     0     0     0     0
0     0     0     0     0
```

```
1     1     1     1
1     1     1     1
1     1     1     1
```

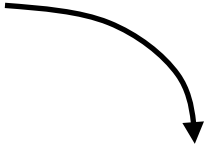
```
20    21    22
 1     2     3
 4     5     6
 7     8     9
```

```
15     1     2     3
16     4     5     6
17     7     8     9
```

Accessing elements of a matrix

- Matlab uses parenthesis ( ) to access elements of a matrix
- Indexes start at 1, so the first of matrix mat is `mat(1,1)`
- Example: `a(3,5)=56.8;`

```
0.1737    0.3421    0.6391    0.1632    0.2313
0.7858    0.7742    0.0934    0.2763    0.8453
0.3656    0.1478    0.9288    0.1310    0.7264
0.7769    0.1482    0.4851    0.0232    0.6947
```

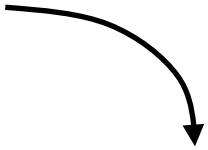


```
0.1737    0.3421    0.6391    0.1632    0.2313
0.7858    0.7742    0.0934    0.2763    0.8453
0.3656    0.1478    0.9288    0.1310   56.8000
0.7769    0.1482    0.4851    0.0232    0.6947
```

Accessing elements of a matrix

- Ejemplo 2: `a([2,3],[2,4])=ones(2,2);`
  o bien: `a([2,3],[2,4])=1;`

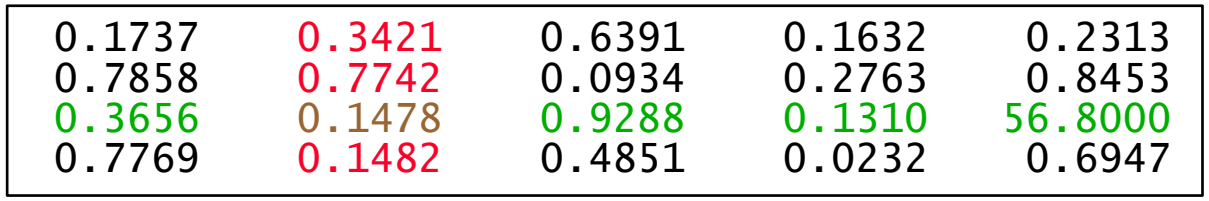
```
0.1737    0.3421    0.6391    0.1632    0.2313
     0         0         0         0    0.8453
     0         0         0         0   56.8000
0.7769    0.1482    0.4851    0.0232    0.6947
```



```
0.1737    0.3421    0.6391    0.1632    0.2313
     0    1.0000         0    1.0000    0.8453
     0    1.0000         0    1.0000   56.8000
0.7769    0.1482    0.4851    0.0232    0.6947
```

Accessing elements of a matrix

- The symbol ' : ' means “all elements”
- It may be used as “all elements in row” or “all elements in the matrix”, etc.



a(3,:)
size(a(3,:)) → [1 5]

a(:,2)
size(a(:,2)) → [4 1]

a(:) → all elements
size(a(:)) → [20 1]
returns a column vector

Basic mathematical operators

- Arithmetic operators: + - * / ^
  - Matlab works with matrices, in contrast with other programming languages that only work with scalar values

```
>> a=rand(2,5);
>> b=rand(5,2);
>> c=a*b;                % matrix 2x2
>> d=b*a;                % matrix 5x5
```

- Operations element-by-element: + - .* ./ .^

```
mn = magic(4);
```

```
 16     2     3    13
  5    11    10     8
  9     7     6    12
  4    14    15     1
```

mn*mn
mn^2

```
345   257   281   273
257   313   305   281
281   305   313   257
273   281   257   345
```

mn.*mn
mn.^2

```
256     4     9   169
 25   121   100    64
 81    49    36   144
 16   196   225     1
```

Other matrix operations

- Sum: `sum`

```
>> b=sum(A);   % if A is a matrix, add elements by column. b is a row vector
>> c=sum(b);   % if b es vector, all elements are added. c is a scalar

>> c=sum(sum(a));  % addition of all elements in a
>> c=sum(a(:));    % addition of all elements in a
```

- Mean and stander deviation: `m=mean(A);` `sigma=std(A);`
- Elements of the diagonal: `v=diag(A);`
- Left division: `x=A\B;` The least squares solution for A x = b is obtained by means of x = A \ b;
- Determinant: `c=det(A);`
- Inverse: `B=inv(A);`
- Eigenvalues: `v=ein(A);`
- Absolute values, or complex module: `B=abs(A);`

Data types

- Matlab uses type **double** according to the IEEE standard
- It can manage special values like inf (infinity) y NaN (not-a-number)
- Complex number are used automatically when needed.

```
>> a=123/0
Warning: Divide by zero.
a =
   Inf

>> b=0/0
Warning: Divide by zero.
b =
   NaN

>> Inf-Inf
ans =
   NaN

>> c=15+sqrt(-1)
c =
  15.0000 + 1.0000i
```

Try it yourself

- Everybody
  - Try different examples to access data and extract data

- Advanced
  - Build a vector containing NaN values. Plot the vector
  - Set axis limit to -inf or +inf

Contents

1. Introduction to Matlab.
2. Basic data types.
3. Matlab programming.
4. Advanced data types.
5. Code Optimization.
6. Graphical representation.
7. Developing standalone applications.

Scripts

- One **script** is a sequence of instructions that matlab can save in a `.m file`

```
%Script de ejemplo

%% Inicio
a=magic(4);
fprintf('Inicio cálculos\n');

%% Traza
traza=sum(diag(a));

%% Resultado
fprintf('La traza vale: %f\n',traza)
```

`ejem_script.m`

- It is executed by the filename:

```
>> ejem_script
```

Functions (Call)

- Matlab functions can receive several arguments and they can also return several results:
  [m,d]=med_des(x);

- Functions can have optional arguments

```
mit=imread('cameraman.tif','TIFF');
mit=imread('cameraman.tif');
```

- It is not necessary to assign all results

```
[mit,map]=imread('imageman.gif');
mit=imread('imageman.gif');
```

Functions (declaration)

- Functions are also written in `.m` files that must me located in the current working directory (or a directory specified in the `path`)

```
function [med,des]=med_des(x)
% Funciona para calcular la media y la desviación a la vez
%   [med,des]=med_des(x)
%
% Rafael Palacios (nov/2004)
med=mean(x(:));
des=std(x(:));
```

`Med_des.m`

This is the information that you get if you type: `help med_des`

Logical expressions

- Relational operators: `~= == > < >= <=`
- Logical operators:

&&	Short-circuit AND
||	Short-circuit OR
&	AND
|	OR

- There is a function for `xor`, but not an operator

Control de Flujo: if

- if

```
if a > b
   tmp=a;
   a=b;
   b=tmp;
end
```

```
if rem(n,2) ~= 0
   M = odd_magic(n)
elseif rem(n,4) ~= 0
   M = single_even_magic(n)
else
   M = double_even_magic(n)
end
```

In contrast with C, Matlab does not requiere parenthesis for the logical expression

Control de Flujo: for

- `for` loop

```
for n = 3:32
   r(n) = rank(magic(n));
end
```

```
a=[];
for n = [ 1 2 3 5 7 11 ]
   a = [a, isprime(n)];
end
```

Control de Flujo: while

- while loop

```
while ~isprime(x)
    x = x + 1;
end
```

Control de Flujo: switch

- switch-case

```
switch (rem(n,4)==0)+(rem(n,2)==0)
    case 0
       M = odd_magic(n)
    case 1
       M = single_even_magic(n)
    case 2
       M = double_even_magic(n)
    otherwise
       error('This is impossible')
 end
```

In contrast with C, Matlab doesn't use `break`.

Control de Flujo: try

- `try-catch`

```
try
   statement
   ...
   statement
catch
   statement
   ...
   statement
end
```

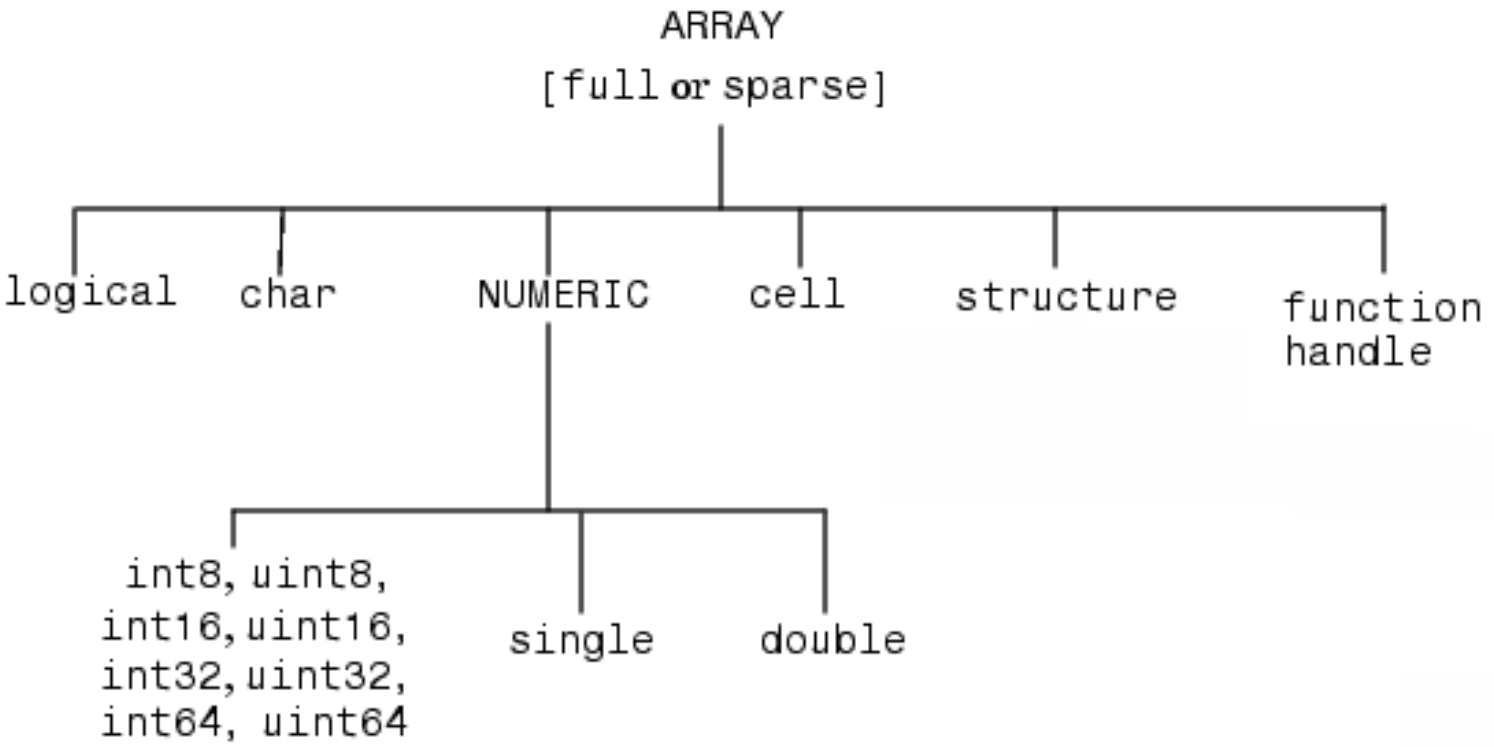
The instructions between `catch` and `end` are only executed in case of error among the first set of statements. One may use `lasterr` to get the error code that triggered the catch section.

Contents

1. Introduction to Matlab.
2. Basic data types.
3. Matlab programming.
4. Advanced data types.
5. Code Optimization.
6. Graphical representation.
7. Developing standalone applications.

All data types

- Matlab has 15 data types that can be used to build matrices or arrays



In addition there are user defined data types for object oriented programming: *user classes*, y *Java classes*

Identifying the type of data

- Decription of the data type

```
>> type=class(x)
type =
double
>>
```

- Logica identification

```
isinteger(x)
isfloat(x)
ischar(x)
islogical(x)
iscell(x)
isstruct(fecha)
```

```
int8, uint8
int16, uint16
int32, uint32
int64, uint64
```

```
single
double
```

```
isempty([])
isinf(Inf)
isnan(NaN)
```

Character strings

- In matlab strings are vectors of character (like in C)

```
>> str='Hello World';
>> whos
  Name      Size                     Bytes  Class
  str       1x11                        22  char array
Grand total is 11 elements using 22 bytes

>> str(7)
ans =
W

>> str=['H', 'o', 'l', 'a'];

>> nombres(1,:)='Rafael';
>> nombres(2,:)='Ana   ';
```

Variables that contain several names, build a matrix of char in which all the names have the same length. The conversion function `char` helps to build such matrix

```
>> names=char('Rafael', 'Ana');
```

Using **cell arrays** it is possible to store several strings of different lengths into one variables

Strings

- Other functions for strings
  - `strrep`: typical find-and-replace

```
st2=strrep(st1,'find','replace');
```

  - `findstr`: find a string within another

```
pos = findstr('find', st);
```

  - `strcat`: concat 2 or more strings

```
text = strcat(st1, st2, st3);
```

  - `sprintf`: builds a string. Equivalent to `sprintf` in C

```
st=sprintf('I have %6.2f EUR',my_money);
```

Structures and Cell Arrays

- Structures allows matlab to store several variables of different types under one variable name

```
>> dot.x=123;
>> dot.y=34;
>> dot.color='red';
>> dot
dot =
         x: 123
         y: 34
     color: 'red'
```

  - Structures do not requiere previous definition
  - Fields access is similar to C
  - It is possible to use vectors of structures

```
>> punto(2).x=435;
```

Structures and Cell Arrays

- A cell array is used to build vectors in which each element may use a different data type:

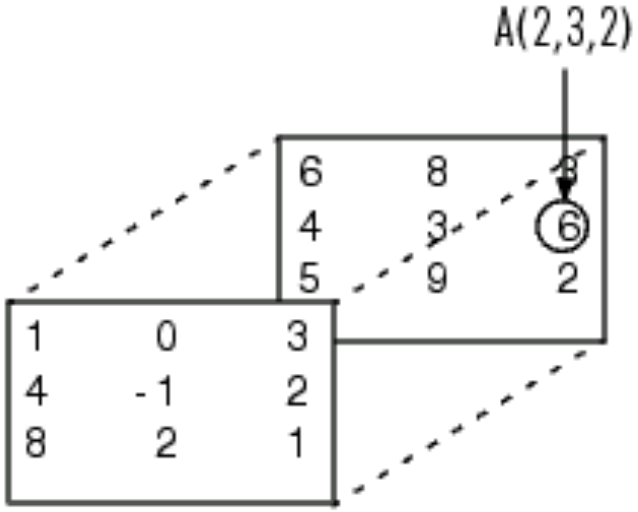
```
>> c={12,'Red',magic(4)};
>> c
c =
    [12]    'Red'    [4x4 double]


>> b{1}=12;
>> b{2}='Red';
>> b{3}=magic(4);
>> b
b =
    [12]    'Red'    [4x4 double]
```

  - One should use { } instead of [ ] or ( )
  - The difference with structures is that you may use indexes instead of field names
  - Structures and cell arrays are less efficient that matrices

Multi-dimensional matrix

- Matrices of more that 2 dimensions are called **Multidimensional Arrays**
- Matlab support all matrix operations in N dimensiones




```
A(:,:,1) =

     1     0     3
     4    -1     2
     8     2     1

A(:,:,2) =

     6     8     3
     4     3     6
     5     9     2
```

Multi-dimensional matrix

```
>> c=imread('autumn.tif');
>> whos c
  Name        Size                          Bytes  Class
  c         206x345x3                      213210  uint8 array

Grand total is 213210 elements using 213210 bytes

>> imshow(c)

>> max(c(:))

ans =

   248

>> gris=(c(:,:,1)+c(:,:,2)+c(:,:,3))/3;

>> imshow(gris)
```

todos los elementos

Try it yourself

- Everybody
  - declare variables of different data types
  - check the sizes of matrices
  - write a short function and call it from matlab

- Advanced
  - Load an image as a 3D matrix
  - Create a cell array

Contents

Measuring time

- Basic functions to measure execution time
  - `tic` & `toc` measure elapsed time in seconds

```
>> tic; inv(inv(inv(randn(1000)))); toc
Elapsed time is 10.015000 seconds.
```

```
tic
   for k = 1:100
      -- programa rápido --
   end
toc
```

  - `cputime` measures CPU time in seconds

```
>> t=cputime; inv(inv(inv(randn(1000)))); e=cputime-t
e =
    9.5137
```

Code analysis

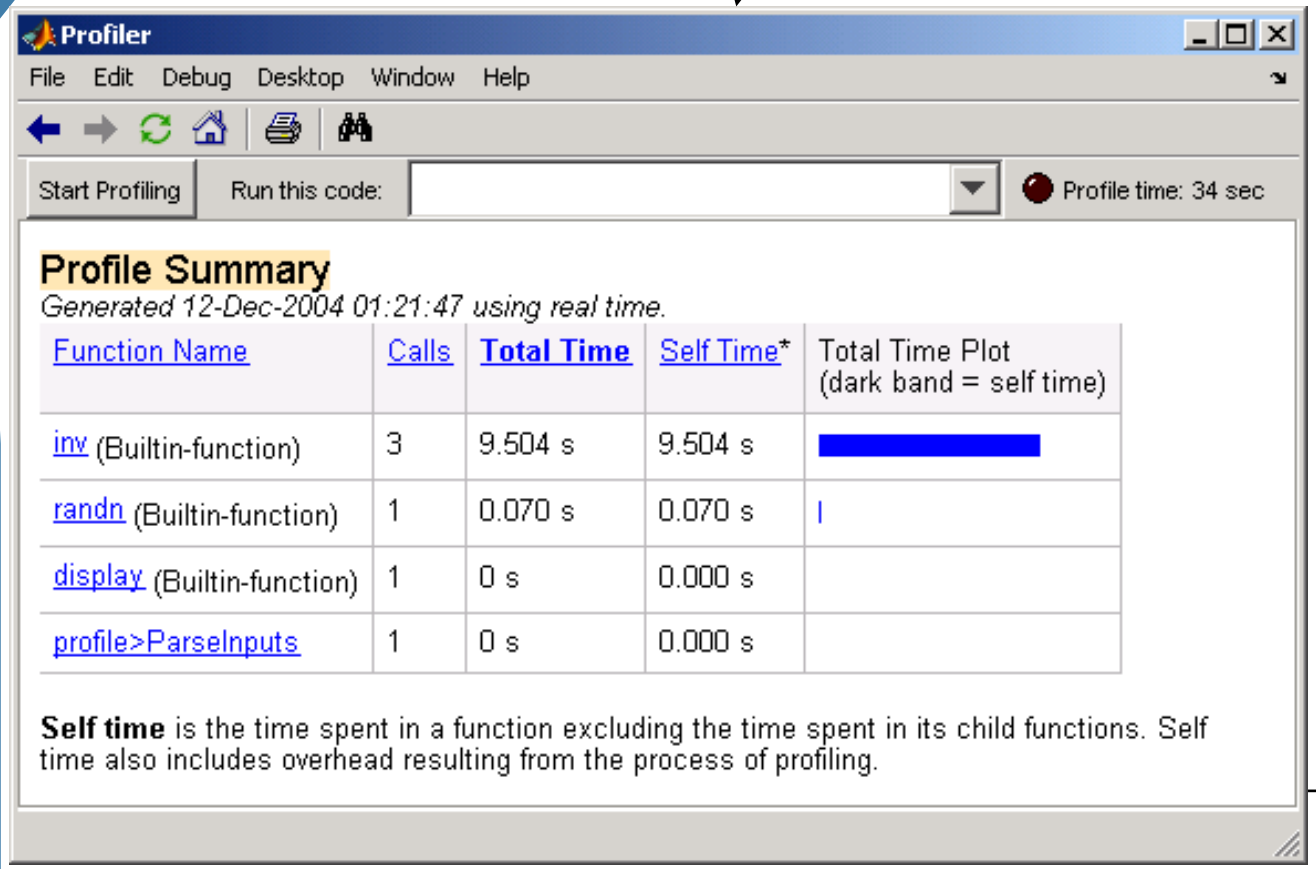
- `profiler` generates a report of the performance of the code

Graphical mode:

```
>> profile viewer
```

Commands

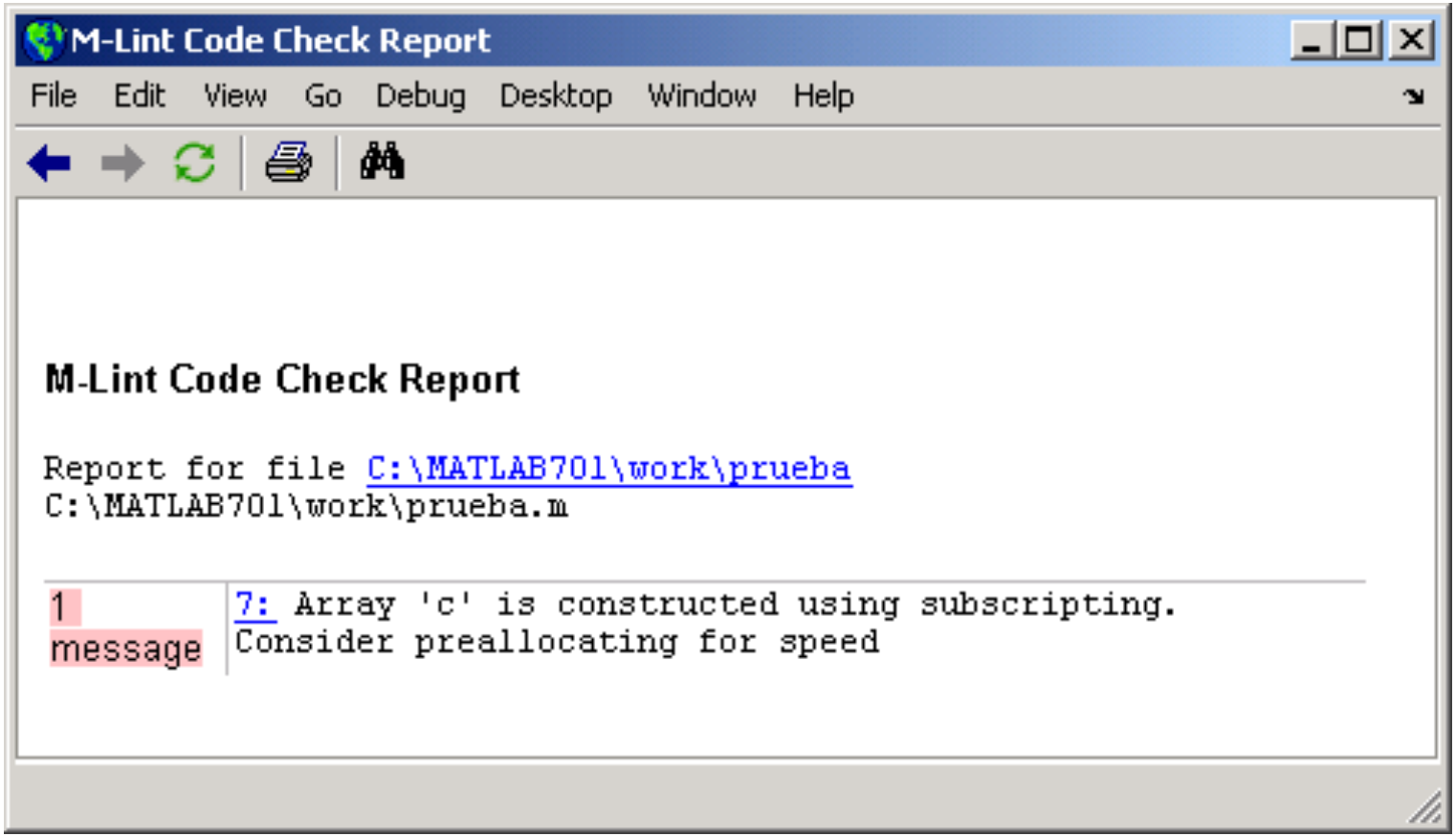
```
>> profile on
>> inv(inv(inv(randn(1000))));
>> profile off
>> profile report
```



Profiler detects the function that it is worth improving.

Code analysis

- `M-Lint` is used to analyze the code and automatically detect possible improvements.



Try it yourself

- Everybody
  - Test your function with M-Lint
  - Use tic and toc to measure execution time

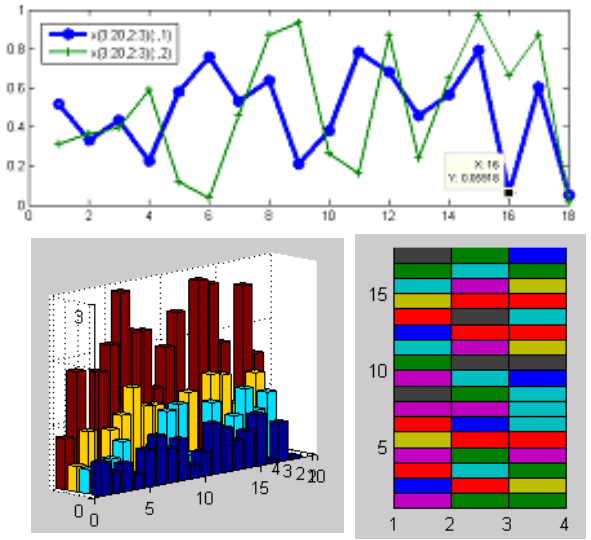
- Advanced
  - Prepare a larger program in which one function calls another.
  - Run your program using profiler
  - Identify bottlenecks.

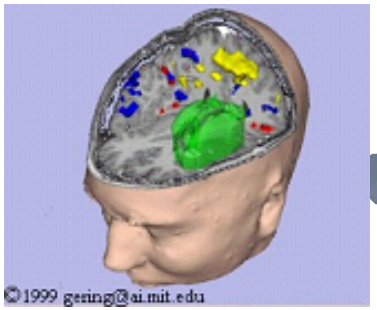
Contents

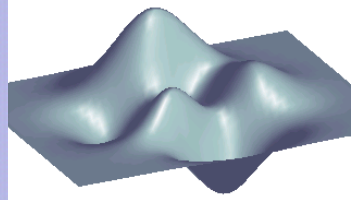
1. Introduction to Matlab.
2. Basic data types.
3. Matlab programming.
4. Advanced data types.
5. Code Optimization.
6. Graphical representation.
7. Developing standalone applications.

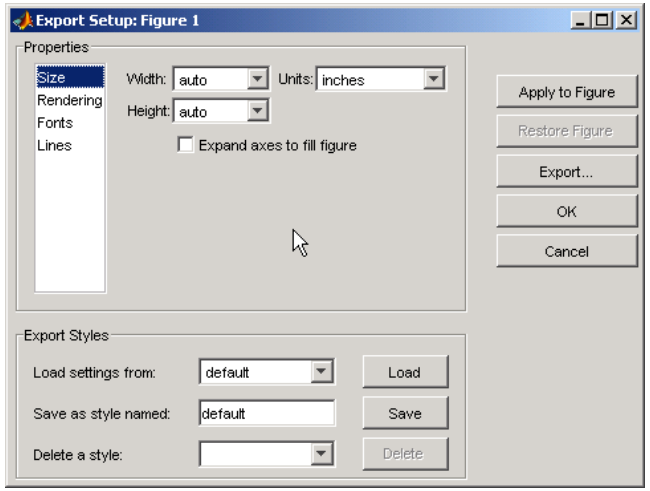
Create graphics

- Matlab can create different types of graphics:
  - to display data values
  - to create images/movies/VR/GIS
  - to generate a graphical use interface



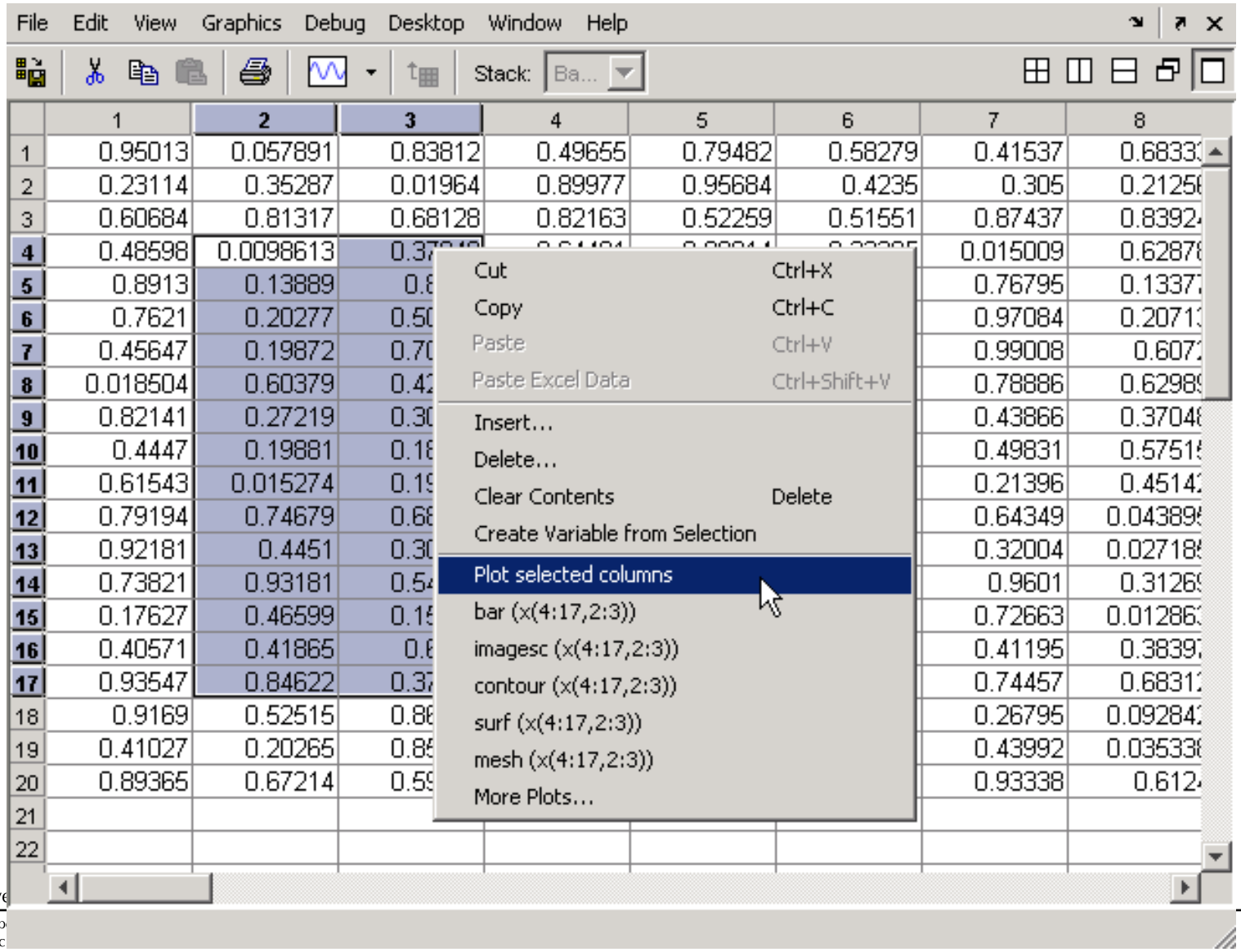




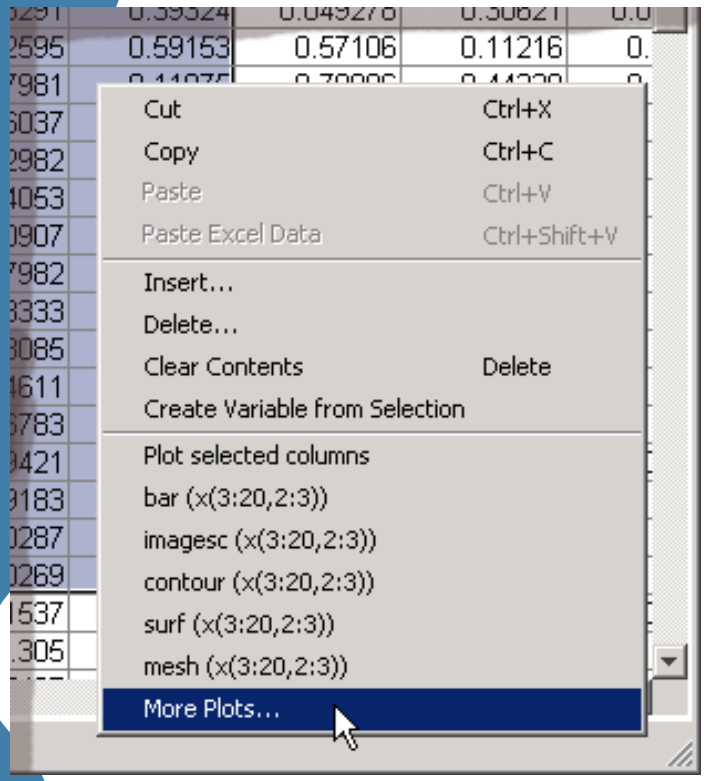


Create graphics

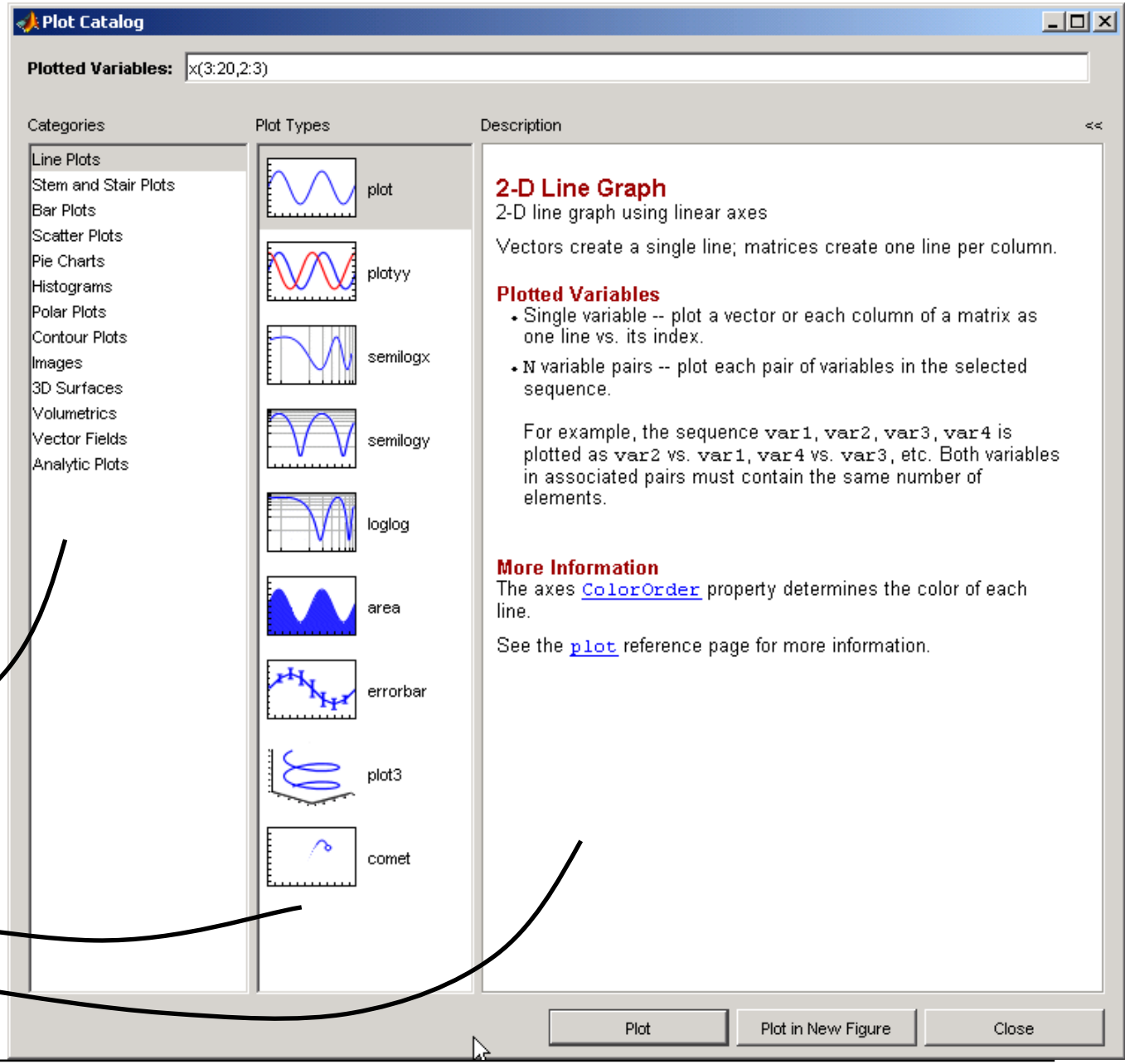
- Create graphics directly from the matrix editor



Select graphics type



Matrix editor



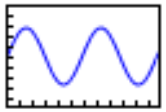
Categories

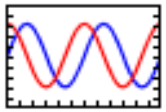
Plot types

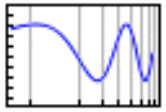
Description and function references

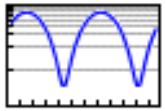
Types of graphs (1D, 2D)

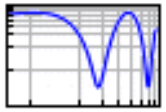
Line

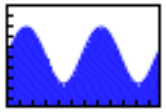

plot

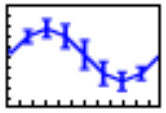

plotyy


semilogx

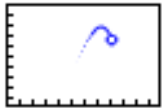

semilogy


loglog

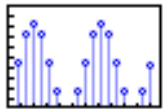

area

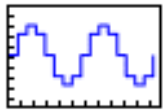

errorbar


plot3


comet

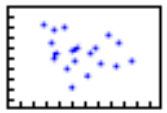
Stem & stair

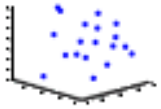

stem

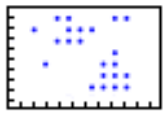

stairs

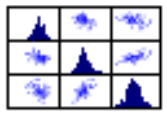

stem3

Scatter

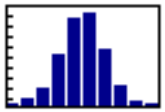

scatter


scatter3


spy

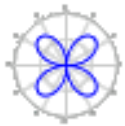

plotmatrix

Histogram


hist


rose

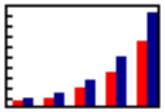
Polar

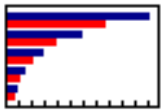

polar

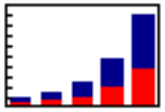

rose

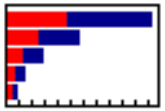

compass

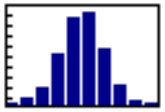
Bar

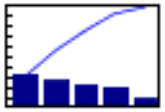

bar

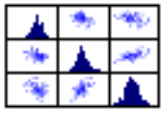

barh


bar


barh


hist


pareto


plotmatrix

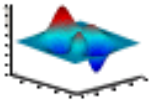
Pie

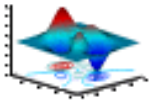

pie

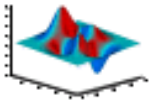

pie3

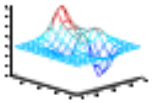
Types of graphs (>=3D)

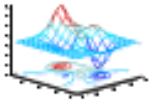
3D surfaces

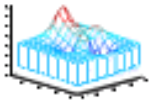

surf

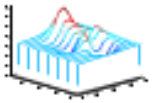

surfc


surfl


mesh


meshc

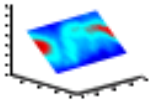

meshz

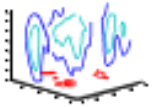

waterfall

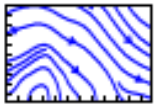

ribbon


contour3

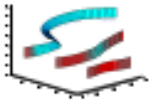
Volumetrics


slice

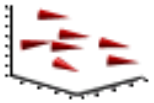

contourslice


streamslice

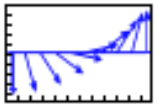

streamline


streamribbon

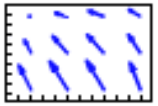

streamtube

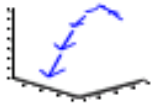

coneplot

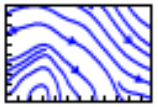
Vector Fields


feather

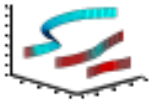

compass


quiver

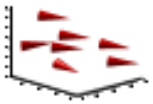

quiver3


streamslice

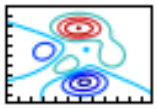

streamline

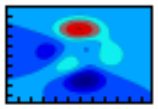

streamribbon


streamtube

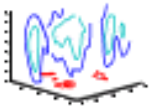

coneplot

Contour

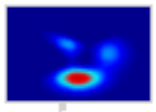

contour


contourf

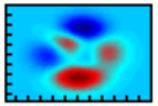

contour3

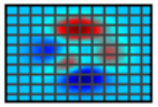

contourslice

Images

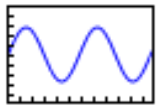


image


imagesc

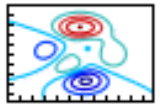

pcolor

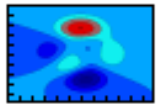
Analytic

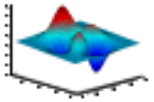

ezplot

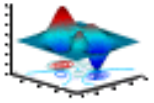

ezplot3

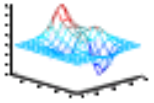

ezpolar


ezcontour


ezcontourf


ezsurf

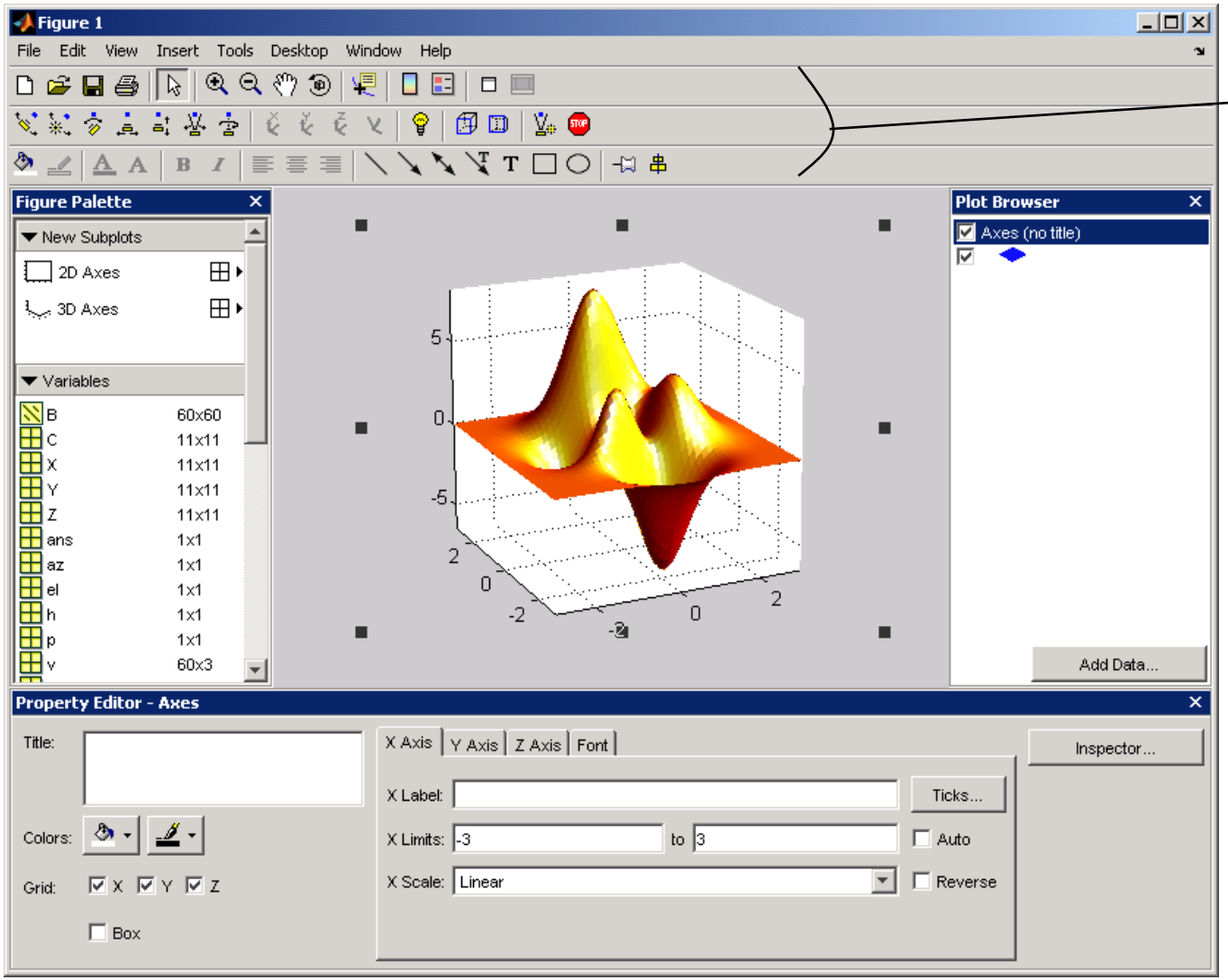

ezsurfc


ezmesh

ezmeshc

Graph adjustments through menus

Ventana de la figura con todas las opciones activadas



Toolbars

Create new subplots

Variables of workspace

Object selector

Properties:
- Figure
- Axes
- Current Object

Save figures

- Using the menu
  - File/Save As → .fig, .eps, .png, .jpeg, .bmp, .pcx, .tiff
  - File/Generate M file

    Esta opción nos permite ver qué comandos se utilizan para crear las modificaciones que hemos realizado por menú

- Using commands
  - `hgsave my_fig` → `my_fig.fig` Can be loaded using hgload
  - save the figure as an image
    - `print -depsc -tiff -r300 archivo`
    - `print –dpng –r150 archivo`

Contents

1. Introduction to Matlab.
2. Basic data types.
3. Matlab programming.
4. Advanced data types.
5. Code Optimization.
6. Graphical representation.
7. Developing standalone applications.

Compiler

- Converts Matlab code to C and generates an executable independent from Matlab
  - The program created does not require a Matlab license to work
  - It may execute faster (not always)
  - Compiler works with functions, not scripts

Installation (only the first time)

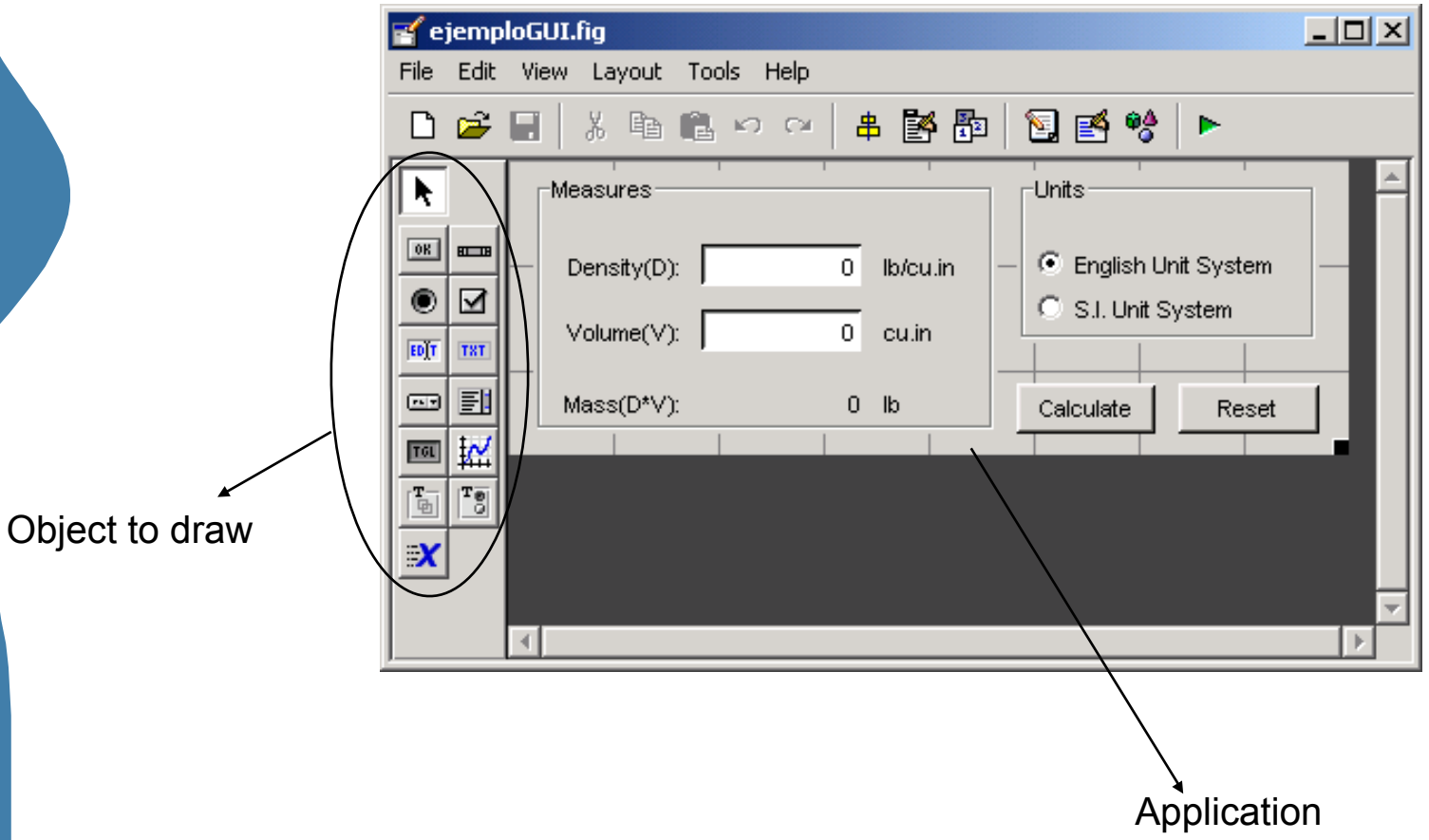
```
mbuild -setup
```

Command to compile

```
mcc -m prueba.m
```

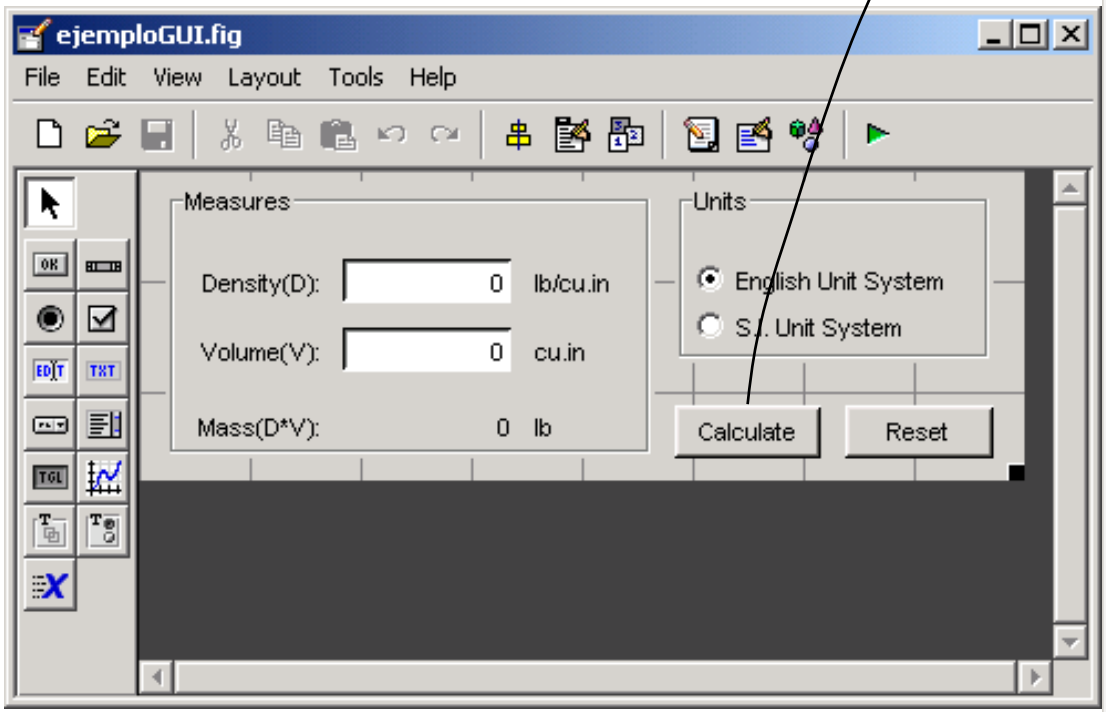
Create graphical user interfaces

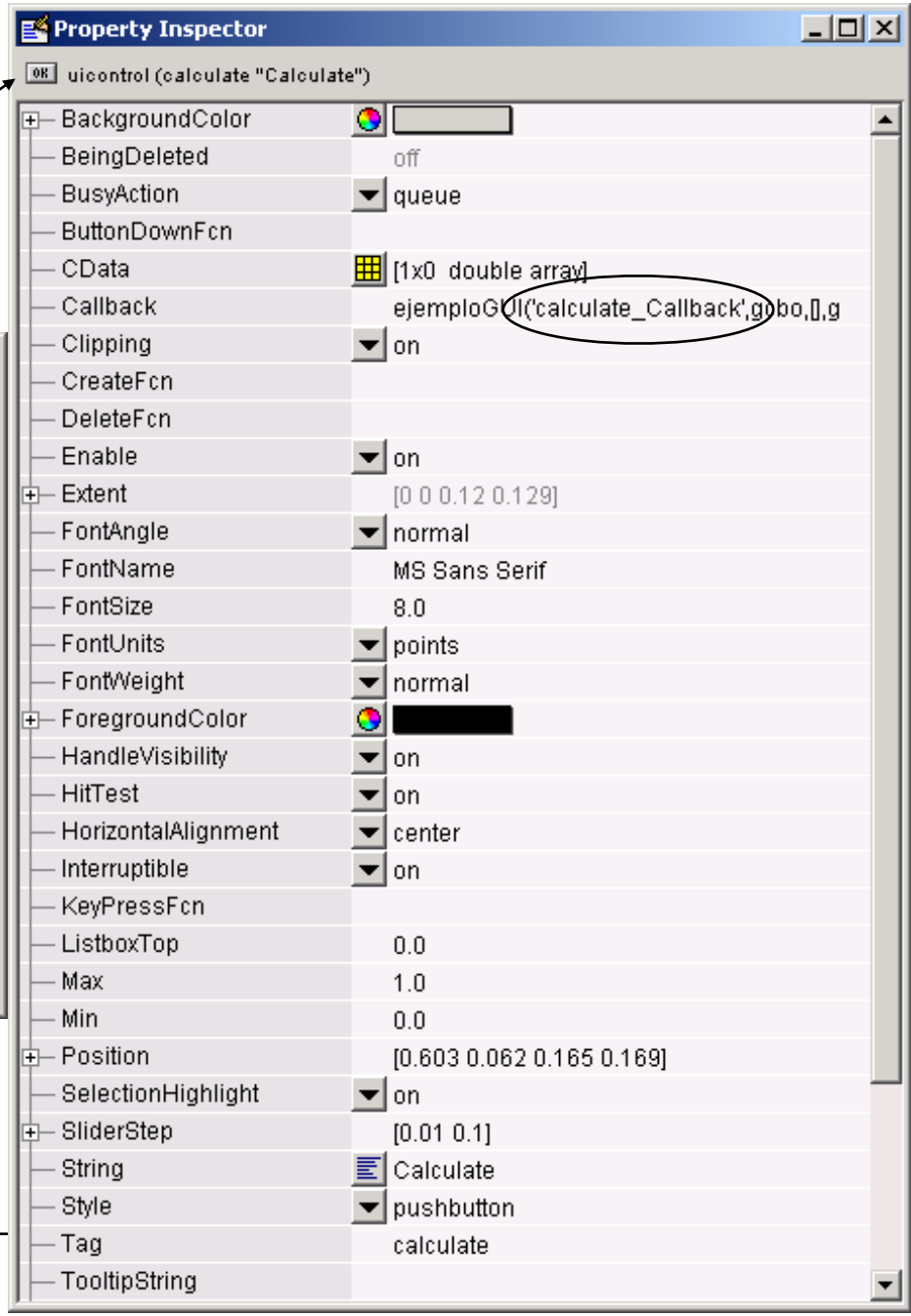
- Use `guide` from Matlab



Create graphical user interfaces

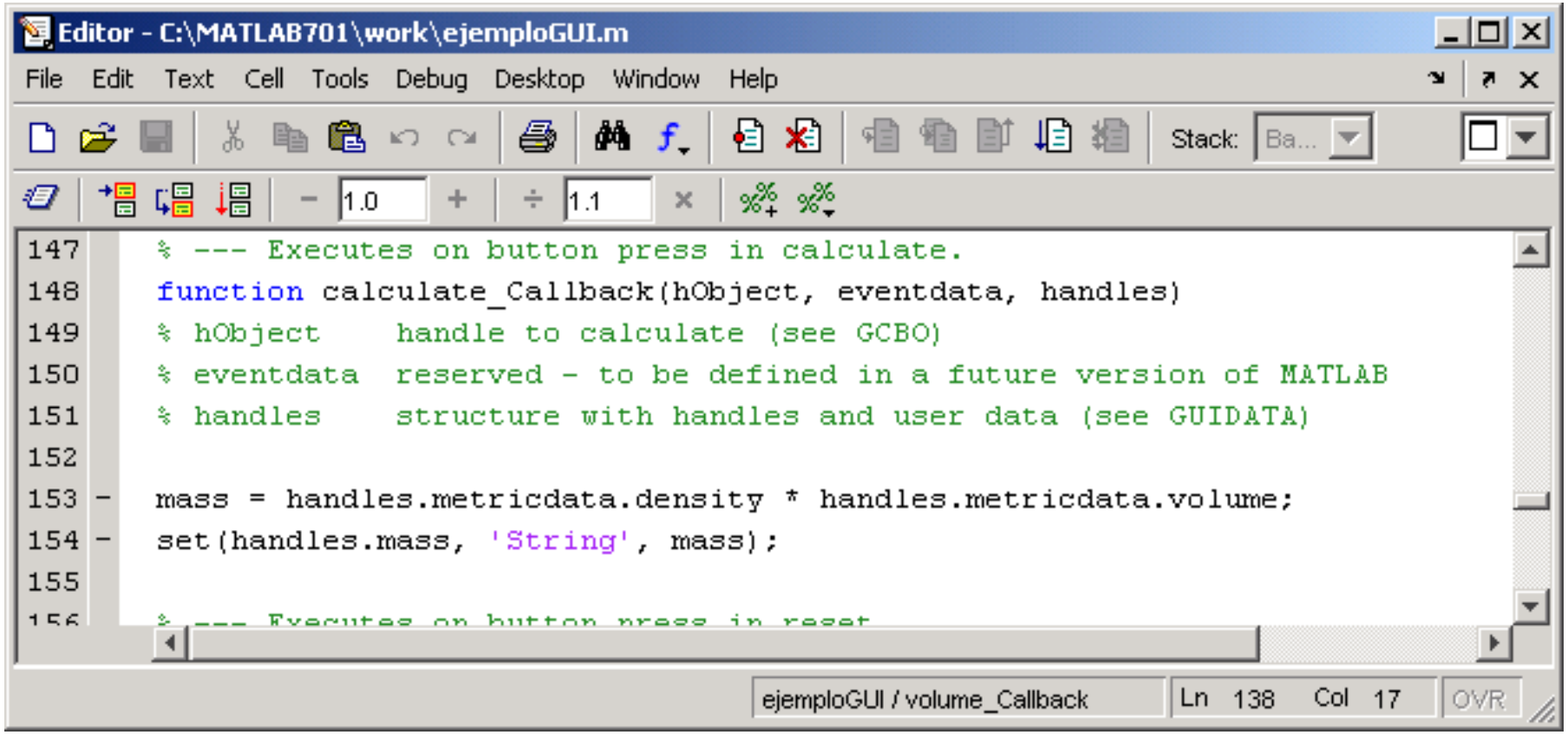
- Each object has some attributes and a callback function





Create graphical user interfaces

- Guide generates a .m file ready to insert your application code



In general it is recommended to keep computing code isolated from interface functions

On-Line resources

- **Mathworks website**
  `http://www.mathworks.com/support/`
  - Documentation
  - Support. Sorted by categories
  - Sample code
  - News
  - Software updates
- **Matlab Central**
  - Newsgroups
  - File Exchange
  - Link Exchange
- **Technical support by email**
  - You need to provide your active license code.
  - You must describe platform, operating system, etc.
  - Be very specific with your problem.

Try it yourself

- Everybody
  - Create a matrix
  - Create a graph
  - Edit graph properties
  - Save you graph, close Matlab, open your graph

- Advanced
  - Play with 3D surfaces
  - Add text to your graph
  - Use Guide to start an application